Applying of the NVIDIA CUDA to the video processing in the task of the roundwood volume estimation

Artem Kruglov[1,*], *Andrey* Chiryshev[1], and *Evgeniya* Shishko[1]

[1] Ural Federal University, 32 Mira street, Yekaterinburg, Russian Federation

Abstract. The paper is devoted to the parallel computing. The algorithm for roundwood volume estimation had insufficient performance so it was decided to port its bottleneck part on the GPU. The analysis of various GPGPU techniques was observed and the NVIDIA CUDA technology was chosen for implementation. The results of the research have shown the high potential of the GPU implementation in the improvement performance of the computation. The speedup of the algorithm for the roundwood volume estimation is more than 300% after porting on GPU with implementation of the CUDA technology. This helps to apply the machine vision algorithm in real-time system.

1 Introduction

Nowadays the GPGPU (General Purpose Computing on Graphic Processing Units) technology has come into widespread acceptance [1]. This programming technique is embedded in the data-flow computing concept. Data-flow computing is the paradigm of parallel computing that interpret data to be processed as a thread which elements could be processed independently hence parallel [2].
Hardware-software architecture CUDA (Compute Unified Device Architecture) developed by the NVIDIA is the implementation of the GPGPU technology. CUDA provides high-level language (C and C++) programming to solve the complex computational problem in a less time due to the multi-core processing power of the GPU [3].

2 The analysis of the parallelization possibility

The R&D project "Hardware-software system for assessment of cubic capacity and type of wood" (grant number 14418/8880 of the FASIE fund) in a part of the software development has resulted in the set of the algorithms for the solving the problem of the roundwood volume measurement via video analysis. All computations referred to the image processing were optimized to running on CPU. The flow diagram of the algorithm is given in the Figure 1.

* Corresponding author: avkruglov@yandex.ru

The sequential implementation gives an insufficient performance of the algorithm thatprevents its implementation in the real-time data processing [4]. The speedupof the algorithm is possible via its adaption to the parallel architecture.

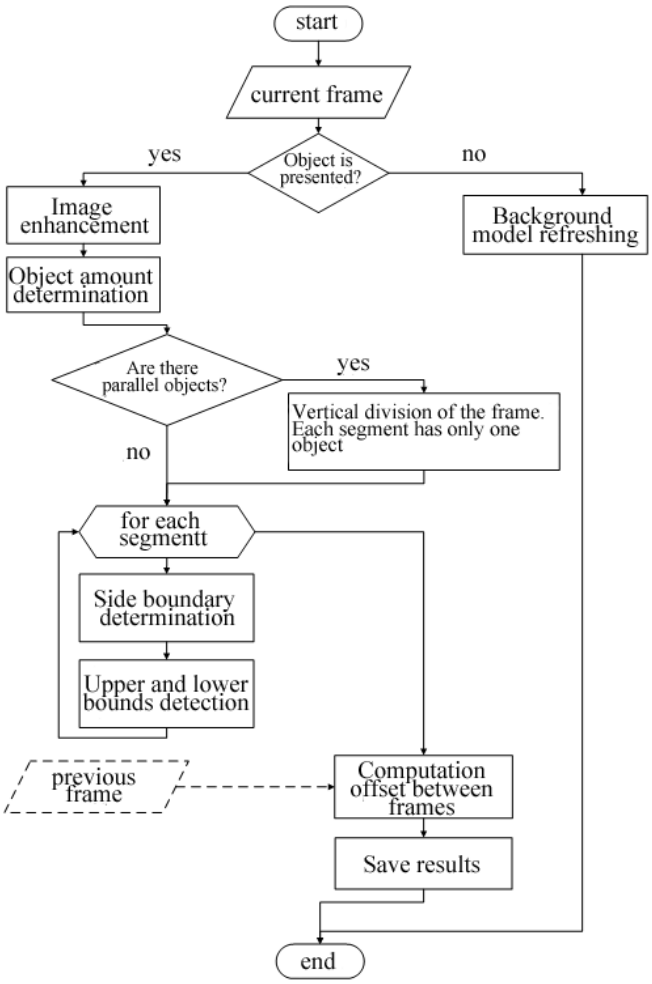
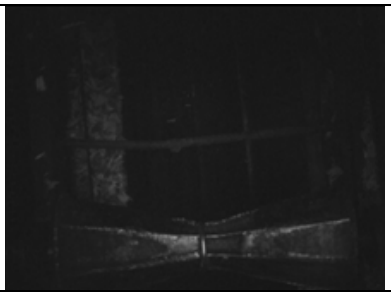
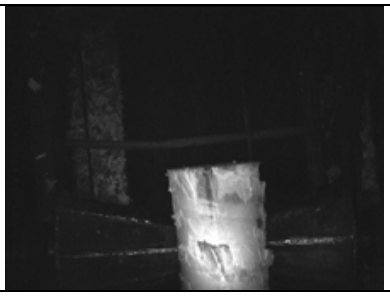
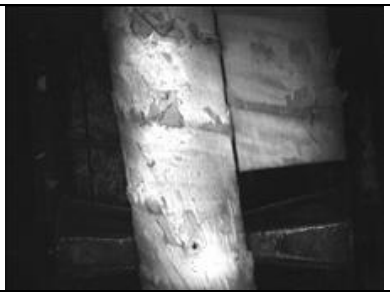
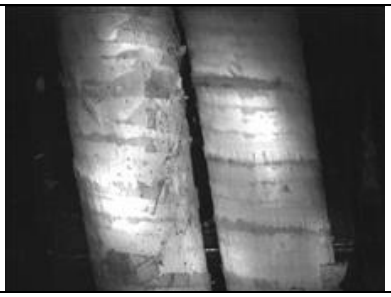


Fig. 1. Flow diagram of the roundwood volume estimation algorithm.

The first step of the algorithm porting on the GPU is its profiling and determination of the time-consuming code areas which need the parallelizing [5]. The complete porting of the algorithm is impossible due to the fact that the GPU has access neither to the memory nor to the input-output devices of the PC inasmuch as it is an auxiliary tool for computing [6].
The computational experiment was carried out to determine the bottleneck of the algorithm. For that purpose a number of the images showing the various cases in the process of the roundwood volume estimation were sampled (Table 1).

Table 1. Set of the images.

№	1	2
Image		
Description	No objects	Appearance of the object
№	3	4
Image		
Description	Tracking of the object	End of the object
№	5	6
Image		
Description	End of the 1 object and top of the 2 object	Appearance of the 2 object while tracking the 1 object
№	7	
Image		
Description	Tracking of two objects	

The algorithm itself was divided into three basic parts:
- image enhancement,
- background model refreshing,
- detection.

The CPU processing time for each part of the algorithm was measured. Results of the computational experiment are given in the Table 2.

Table 2. Results of the computational experiment.

Image	Processing time, ms			
	Image enhancement	Background model refreshing	Object detection	Total
1	0	2,1	0	2,1
2	31,4	0	10,6	42,0
3	32,0	0	11,5	43,5
4	31,9	0	12,0	43,9
5	32,1	0	13,0	45,1
6	31,9	0	13,6	45,5
7	32,2	0	13,4	45,6

The experiment showed that the bottleneck of the algorithm is image enhancement. This operation consumes about 70% of the time of the frame sequence processing. The image enhancement consists of the combination of the basic morphology operators which are the erosion and dilatation. The consequent execution of these methods of the non-linear signal processing is an extremely effective approach to noise reduction within the developed algorithm [7, 8, 9].

The mentioned morphology operators are referred to the group of the local transformation and implemented in sliding mode by successive displacement of the scanning area which includes an odd number of image samplings. All pixels occurring in the scanning area are processed by the specific scenario [10]. The result of the processing is a output image pixel referred to the center of the scanning area. The final value of the pixel is computed as a minimum value among all pixels in the area for the erosion and a maximum value for the dilatation [11]. Since each pixel is processed by the same consequence of the operation (minimum and maximum search), the algorithm could be effectively implemented on the GPU [3, 12].

3 The implementation of the algorithm on GPU

The structure of the image enhancement algorithm using CUDA consists of the following steps:

1. Selecting of the active GPU;
2. Allocating the data store for an image in the global memory;
3. Loading input image into the global memory;
4. Loading coefficients of the filter window into the constant memory;
5. Forming the structure of the computational grid;
6. Launching the kernel functions executing the image filtration;
7. Copying the output image from the GPU memory to the internal memory;
8. Deallocating.

The code is written on CUDA C and consists of the sections which executed on both the CPU and GPU [13]. The host-code with a description is listed below.

```
1   #define BLOCK_DIM 16
2   __constant__ int c_mask[25];
3   // Host-function
4   void FilterGPU (byte *inbuf, byte *outbuf, int height,
int width, int apert, int *mask)
5   {
6   // Device initialization
7   cudaSetDevice(0);
8   // Memory allocating
9   byte *buf;
10  cudaMalloc ((void **)&buf, sizeof(byte)*width*height);
11  // Loading image into the device
12  cudaMemcpy (buf,inbuf, sizeof(byte)*width*height,
cudaMemcpyHostToDevice);
13  // Loading coefficients of the filter window into the
constant memory
14  cudaMemcpyToSymbol («c_mask», mask,
sizeof(int)*apert*apert);
15  // Computational grid forming
16  dim3 blocks = dim3(width/BLOCK_DIM+1,
height/BLOCK_DIM+1);
17  dim3 threads = dim3(BLOCK_DIM, BLOCK_DIM);
18  // Kernel launch
19  kernel <<<blocks, threads>>> (buf, height, width, apert);
20  cudaThreadSynchronize();
21  // Uploading output image to the PC memory
22  cudaMemcpy (outbuf, buf, sizeof(byte)*width*height,
cudaMemcpyDeviceToHost);
23  // Device deallocating
24  cudaFree(buf);
25 }
```

4 Comparison of the approaches

The parallel implementation of the image enhancement was implemented to optimize the algorithm of the roundwood volume estimation. The computational experiment was carried out to compare the performance of the parallel implementation on GPU against the sequential implementation on CPU and OpenMP multithreading technology. The three versions of the algorithm were tested on the set of the images. The computational experiment has been carried out on the IBM PC with the following characteristics:

- 8-core CPU Intel Core i7 920 2.79 GHz;
- GPU NVIDIA GeForce GTS 450 (192 CUDA kernels);
- 64x Windows 7 OS.

Compilation of the parallel and sequential versions was implemented in Microsoft Visual Studio 2013, plugin NSight Visual Studio Edition v3.2 and NVIDIA CUDA Toolkit v5.5 were used for parallel implementation. Results of the experiment are presented in the Table 3.

Table 3. Results of the computational experiment.

Image	Average processing time, ms			Speedup (serial)	Speedup (OpenMP)
	Serial	OpenMP	CUDA		
1	2,1	2,1	2,1	1	1
2	42,0	19,9	12,2	3,44	1,63
3	43,5	21,0	12,8	3,4	1,64
4	43,9	21,8	13,9	3,16	1,57
5	45,1	22,9	14,5	3,11	1,57
6	45,5	22,8	15,2	2,99	1,5
7	45,6	22,8	15,4	2,96	1,48
Average (with image enhancement)				3,17	1,56

As can be seen from the Table 3, the average frame processing speedup is more than 300%. Although the average data transfer time to/from GPU is 0,2 ms, the achieved speedup value clearly demonstrates the efficiency of computing on the GPU.

5 Conclusion

The following problems were solved while porting the existent sequential algorithm on the GPU:
- The existing algorithm for roundwood volume estimation was analyzed for the most effective paralleling on GPU;
- The parallel implementation of the algorithm based on the CUDA technology was offered;
- The performance of the algorithm is improved in three times due to the parallelization. This allows to apply the algorithm for the roundwood volume estimation in the real-time systems.

Thus the results of the research have shown the high potential of the GPU implementation in the improvement performance of the computation. The speedup of the ported algorithm provides the resources for additional functionality in the developed hardware-software system of the roundwood estimation. It was decided to use two color digital video cameras which will enable to analyze the major part of the log surface hence support an external estimation of the timber quality.

References

1. D. Kornev, A. Tarhanov, Image processing by the GPU via CUDA technology. In: Proceedings of the 40th Regional Youth Conference, series “Problems of the theoretic and applied mathematics”. pp. 406-410 (2009)
2. A. Boreskov, A. Harlanov, Fundamentals of CUDA technology, DMK Press (2010)
3. Nvidia CUDA C programming guide (2015)
4. A.V. Boreskov et al., Parallel computing on GPU. CUDA architectural and programming model. In: Moscow Univ. Proceedings, series “Supercomputer Educations”, p. 336 (2012).
5. L. Dorosinsky, V. Kruglov, N. Papylovskaya, A. Chiryshev, CUDA technology in digital image processing tasks. The success of modern science, vol. 10,pp. 88-98(2011)
6. V. Kruglov, N. Papylovskaya, A. Chiryshev, Benefits of sharing CPU and CUDA-device, Fundamental research, vol. 8 № 2,pp. 296-304 (2014)

7. A. Kruglov, V. Kruglov, A. Chiryshev, U. Chiryshev, Implementation of the resource-intensive machine vision algorithms in real-time systems. Fundamental research, vol. 10 № 12,pp. 2612-2619 (2013)
8. R. Tomakova, Hard-segmented image processing method using multi-layer morphological operators. In: Proceedings of the Southwestern State Univ, vol. 2 pp. 158-164 (2012)
9. V. Chernoysov, A. Savchenko, Morphological moving object detection algorithm on noisy video. International scientific-technical conference Information systems and technologies (2014)
10. F. Kornilov, Search structural imaging differences: Algorithms and methods. Machine Learning and Data Analysis, vol. 1 № 7, pp. 902-919 (2014)
11. T. Volosatova, A. Marchenkov, N. Chichvarin, Research and development of the combined method of detection and identification of moving objects. Information Technology, vol. 12, pp. 24-31 (2013)
12. R. Fernando, D. Kirk, GPU gems: Programming techniques, tips and tricks for real-time graphics (2015).
13. T. R. Halfhill, Parallel processing with CUDA (2008).